Tecnológico de Costa Rica

Introducción a la programación

Examen I

Tiempo 1 hora con 45 minutos

Instrucciones:

- Se le presentan 4 problemas para los cuales debe desarrollar en Python los programas que los solucionen. Los 4 problemas son sobre algoritmos numéricos.
- **No puede utilizar listas** en ninguna de las soluciones.
- **No puede convertir** los números **a string**.
- Cada función debe retornar lo que se le solicita, **ninguno debe ser con prints o inputs**.
- Cada ejercicio tiene un valor de 25 puntos del examen.
- Durante los **primeros 30 minutos podrá hacer consultas** sobre el examen a los profesores o asistentes. Pasado este tiempo no se podrá consultar más.
- Utilice los nombres de la funciones **exactamente** como se le solicitan, en caso contrario no se le revisará el ejercicio.
- El uso de Internet está limitado exclusivamente para descargar y subir el examen del TEC Digital. El uso de internet, chats, modelos AI, Copilotos implicarán la cancelación de la prueba y obtendrá una calificación de cero, además del proceso disciplinario que el TEC indica para los casos de plagio.
- Para esta prueba NO podrá consultar algoritmos, programas o notas hechas en clase o de otras fuentes. Solo podrá acceder a hojas en blanco, lápiz y la computadora del laboratorio.

Entrega:

- Debe subir un archivo .py con la solución a los ejercicios. El nombre del archivo debe ser **Ex1_Apellido_Nombre.py**, ejemplo Ex1_Mora_Diego.py.
- La solución la subirá en la sección de Evaluaciones, en Exámenes > Examen I.
- El Tec Digital se cierra a las 9:20 a.m., no se podrán entregar exámenes pasado ese tiempo.

eliminar_repetidos (num)

1. Escriba un programa en Python llamado **eliminar_repetidos (num)** que recibe un número entero positivo o cero y retorna un número sin dígitos repetidos del número dado, dando prioridad de aparición en el resultado a los dígitos menos significativos, es decir, recorriendo el número de derecha a izquierda y eliminando los repetidos más siginificativos en el número dado.
Nota: En el resultado de este programa podría perder el dígito cero y estaría bien. (ver tercer ejemplo)

Ejemplos:
> eliminar_repetidos (135232) retorna 1532
> eliminar_repetidos (555) retorna 5
> eliminar_repetidos (102142) retorna 142 #en este caso pierde el cero y está bien

---

interseccion (num1, num2)

2. Escriba un programa en Python llamado **interseccion (num1, num2)** que recibe 2 números enteros mayores o iguales que cero y determina otro número conformado por los dígitos del primer número que están contenidos en el segundo número. En el resultado no se debe presentar dígitos repetidos. El orden de los dígitos del resultado no es importante. Si no hay intersección retorne -1

Ejemplos:

> intersección (241, 42542) resultado es 24 ó 42
> intersección (30241, 425420) resultado es 204 ó 420 ó 402 ó 240, etc. En este caso no podrá ser 024 o 042 porque perdería el cero.
> intersección (123, 0) resultado es -1
> intersección (123, 89) resultado es -1

---

división_espejo (num)

3. Escribir un algoritmo llamado division_espejo (num) que reciba como argumento un número entero positivo o cero y retorne True si todos los dígitos más significativos son divisibles entre los menos significativos. Si el número tiene cantidad de dígitos impar, el dígito de en medio puede no dividirlo o dividirlo por sí mismo, es irrelevante para el resultado en las dos opciones. *** No puede invertir el número para solucionar.

Dos ejemplos:

Izquierda_vs_derecha (num)

5. Haga un programa en Python llamado **izquierda_vs_derecha (num)** que recibe un número entero positivo o cero y retorna un string ’izquierda’ o ‘derecha’ o ‘empate’. El objetivo del programa es que indique cuál lado del número tiene más números mayores que el otro lado pero comparando el más significativo con el menos significativo, el segundo más significativo vs. el segundo menos significativo y así en sucesión hasta llegar al centro. Al final, si la derecha (menos significativos) tenía más dígitos mayores que la izquierda, la respuesta será ‘derecha’, si la izquierda (más significativos) tenía más dígitos mayores que la derecha, la respuesta será ‘izquierda’. Si ambos lados del número tienen la misma cantidad de mayores respecto con el otro lado, retorna empate. Al comparar dígitos, si son iguales no suma a ningún lado o suma a ambos (su decisión). En caso de cantidad impar de dígitos, el centro no se compara. No puede invertir el número, tampoco convertir el número a string o lista. Ejemplos:

> izquierda_vs_derecha (349503) **retorna** ‘izquierda’

Compara 3 con 3, son iguales
Compara 4 con 0, izquierdo mayor
Compara 9 con 5, izquierdo mayor
El lado izquierdo tiene 2 mayores y el derecho 0,
por tanto el resultado es izquierdo

> izquierda_vs_derecha (7119221) **retorna** “derecha” (1 mayor en izquierda, 2 mayores en derecha)

> izquierda_vs_derecha (4853529) **retorna** “empate” (1 derecha, 1 izquierda y 1 empate)
